NANYANG TECHNOLOGICAL UNIVERSITY

DISTILLATION AND SELF-TRAINING IN LANE DETECTION

Ngo Jia Wei

School of Computer Science and Engineering
2020

NANYANG TECHNOLOGICAL UNIVERSITY

SCE19-0563

DISTILLATION AND SELF-TRAINING IN LANE DETECTION

Submitted in Partial Fulfillment of the Requirements
for the Degree of Bachelor of Engineering (Computer Science)
of the Nanyang Technological University

by

Ngo Jia Wei

Acknowledgement

Special thanks to my Final Year Project supervisor Prof Chen Change Loy for his support, supervision and guidance throughout the project and for allowing me to conduct research on topics which deeply interest me on a personal level.

Special thanks to my mentor, Mr Wenwei Zhang for his support, supervision and guidance throughout the project as well as our weekly discussions where he shares his knowledge to help me overcome problems and possible directions I may take.

Special thanks to Prof Shijian Lu for taking the time and effort to assess my research and report. Last but not least, I would like to thank everyone who has supported, inspired and given me the motivation for the entire course of this project.

Contents

List of Figures

List of Tables

Distillation and self-training in lane detection

Jia Wei, Ngo
Nanyang Technological University
ngoj0003@e.ntu.edu.sg

Abstract

Techniques such as knowledge distillation and self-training have seen much research in recent years. These techniques are generalisable and provide performance improvements when applied on most models. Distillation allows a student network, usually with a smaller capacity, to perform similarly to their larger teacher networks, while retaining its lightweight and fast properties. Self-training allows us to utilize unlabeled images at scale to improve our network's performance. Existing research has seen experimentation mainly on classification tasks, with some recent papers exploring distillation and self-training in the semantic segmentation domain, but to the best of our knowledge, never simultaneously. In this paper, we set out to explore the performance gains that can be achieved from these techniques in the domain of lane detection for self-driving cars. Our results show that Knowledge Distillation with dark knowledge from an ensemble of same architecture models will be able to provide similar performance gains as with ensembling techniques, while retaining its low evaluation time compared to ensembling techniques (an important factor for lane detection in self-driving cars). Preliminary results from self-training, which has seen positive results when used in conjunction with pre-training, shows we may be able to provide additional performance gains on top of ensemble distillation for lane detection with large amounts of unlabeled data.

1. Introduction

Lane detection [5] for self-driving cars is a crucial task. It provides lateral bounds for the movement of the vehicle, which is critical for road safety, and an essential component of self-driving cars.

Traditional methods for lane detection extracted low-level features of lanes using hand-crafted features like colour and edge. These features are then combined with techniques like Hough transform [12] and Kalman filter [7]. These combined features would then generate lane segment information. These traditional methods are simple, but more subsceptible to various environmental challenges such as the lighting conditions and occlusion. It is also prone to scalability issues due to different road scene variations.

Recent developments in deep learning have seen widespread adoption of Deep Neural Networks (DNN) for the task of lane detection. Lane detection is generally approached as a semantic segmentation problem. However, these methods do not make use of global and spatial properties provided by the whole image. Several deep learning methods have been introduced [22, 17, 24] to utilize these structural information in order to solve challenges such as occlusion and to make use of global information.

DNNs are highly flexible nonlinear algorithms which are capable of learning a near infinite number of mapping functions. As such, training multiple networks, even with the same settings and dataset, will yield different performances every time. DNNs are complex and can take hours, weeks or even months of training to converge depending on its complexity. We frequently train multiple models to achieve a 'best' performing model, which causes our efforts and resources used on all other models to be 'wasted'.

One solution is to utilize ensembling learning, introduced in [11]. Ensembling learning has been garnering much attention over the past decades and has been shown to achieve better performances when utilizing predictions from an ensemble of multiple models as compared to a single best network used in isolation [6]. It is still frequently seen being used as top solutions in Kaggle competitions.

With ensemble learning, we are able to make use of all our pre-trained models to provide a performance boost and solve our problem of 'wasted' resources in training. However, one of the drawbacks often overlooked is the increased evaluation time overhead from multiple models being used to predict the model's output instead of just one. With limited memory or computational power in portable devices used in cars for lane detection, predictions will be slow and unable to achieve real time. For a time sensitive task such as lane detection, this is not ideal.

Hence, we propose utilizing our multiple pre-trained same-architecture models as an ensemble of teachers to perform distillation to achieve performance gains from ensem-

ble learning while retaining an evaluation time capable of real time. Previous research has been done in compressing large, complex model ensembles into a smaller, faster model without significant loss in performance [10]. However, additional unlabeled data is used for the compression. Hinton *et al*. experimented with distillation from an ensemble of models and noted that using the original train data was still beneficial, but experiments were mainly in the domain of classification.

In this paper, we set out to explore model compression through distillation from our ensemble of same-architecture teachers without introducing any new data, in the context of lane detection.

Self-training allows us to utilize unlabeled data at scale to improve the network's performance. Self-training is one of the common solutions for task domains where it is difficult to obtain labeled data [4]. Self-training has also seen research for tasks domains which have a plethora of unlabeled images which are easy to obtain. Xie *et al*. [32] utilized self-training to achieve performances which exceeded previous state-of-the-art classification on ImageNet. With great interests in self-driving cars in recent years, unlabeled images of lanes have become abundant and easy to obtain. As such, inspired by [32], we explore possible performance gains from self-training on the lane detection domain by utilizing unlabeled images at scale.

In summary, our contributions are: (1) We conduct novel experimentation of Knowledge Distillation from an ensemble of same-architecture teachers in the context of semantic segmentation and lane detection, where previous studies mainly focused on classification or included additional unlabeled data. (2) We combine self-training with Knowledge Distillation, which to the best of our knowledge has never been done simultaneously, with previous research mainly focused on self-training concurrently with pre-training or strong data augmentation as in [32, 38]

2. Related Work

Lane detection. Lane detection was regularly handled with hand-crafted features to obtain lane segments. These segments are further grouped to get the final result [12, 7]. As mentioned, these traditional methods are more subsceptible to environmental factors and only applicable to relatively easy scenarios. Recently, developments in deep learning have omitted hand-crafted features completely to extract features in an end-to-end manner. These approaches generally treat lane detection as a semantic segmentation problem, where each pixel in an image is classified to whether it is a lane or not. Since lanes are inherently long and thin, the ratio of lane pixels to background pixels are usually very small and is as such a challenging task.

Convolutional Neural Networks (CNN) have been widely adopted for a wide variety of visual perception tasks. The domain of lane detection has also seen adoption of CNNs being applied to semantic segmentation [1, 9, 21]. Recently, exceptional results have been obtained by CNNs for semantic segmentation [19].

Knowledge distillation. Knowledge distillation was first introduced by [10, 14] to transfer knowledge from larger cumbersome networks into lightweight smaller networks. The main goal of knowledge distillation is to teach the smaller student network to mimic the large teacher network in order to improve the student network's performance while still maintaining its lightweight and quick runtime (*model compression*).

Recent studies have investigated learning from an ensemble of teacher networks [2, 13, 20, 30, 33] and have found that multiple teachers can combine different predictions to attain a more extensive understanding of the knowledge, resulting in a more integrated dark knowledge for the student [35]. This would allow us to absorb the performance gains from an ensemble of teacher models and condense it into one student model.

Self-training. Self-training [28, 34, 25, 26] is a technique which first uses labeled data from a dataset to train a teacher model, and subsequently uses this teacher model to generate pseudo-labels on unlabeled data. The pseudo-labels generated from this teacher model are then used in conjunction with the initial labeled dataset to jointly train a new student model. Typically, this process can be further iterated by using the newly trained student model as the teacher model to generate new pseudo-labels on unlabeled data and using the new pseudo-labels with the original dataset to jointly train a new student model k times.

Recent studies [32, 38] have shown that performing augmentation and adding noise to the student network will allow the student to learn beyond the teacher's knowledge and provide even further performance gains.

Other existing methods in semi-supervised learning included consistency training [31] and pseudo labeling [3]. However, these methods do not have a separate teacher model and used the model currently being trained to produce pseudo labels.

In the domain of semantic segmentation, Babakhin *et al*. [4] utilized self-training to improve the performance on previous networks, showing that self-training is also effective in the domain of semantic segmentation.

Self-training provides a simple and effective algorithm to leverage unlabeled images at scale to improve network performances. Self-training is also one solution for domain tasks with scarce or difficult to label datasets. For a domain such as lane detection where unlabled videos and images are abundant and relatively easy to acquire, self-training seems a viable and simple solution to improve performances.

3. Methodology

As mentioned, lane detection is commonly approached as a semantic segmentation task. The objective of semantic segmentation is to assign labels $l_{i,j}, (l_{i,j} = 1, ..., N_c)$, where N_c is the number of classes, to each pixel (i, j) in an input image $\mathbf{X}$, to a segmentation map s. In other words, the task is to learn a function mapping of $f : \mathbf{X} \rightarrow s$. Following recent studies such as [15, 22] which introduce the task of lane existence prediction using binary labels to indicate the existence of lanes to facilitate the process, our task becomes $f : \mathbf{X} \rightarrow (s, b)$, where b are binary labels to indicate the existence of lanes.

3.1. Ensemble learning

After training our lane detection network for the previously mentioned semantic segmentation task for multiple times in search of an optimal model, we perform ensemble learning on all our trained networks. There currently exist many different ensembling methods such as Max Voting, Averaging, Weighted Average, Stacking, Blending and Boosting. We use a simple prediction averaging. For a set of trained models $t = \{1, ..., T\}$, the ensemble prediction estimate is,

$$\bar{f} : \mathbf{X} \rightarrow (s, b) = \sum_{t=1}^{T} \frac{1}{T} f_t(\hat{s}_t, b_t) \tag{1}$$

where $\hat{s}_t$ is the standard softmaxed (temperature=1) prediction output of the segmentation map s_t from f_t, $t = \{1, ..., T\}$.

3.2. Knowledge Distillation from an ensemble of Teachers

The intuition behind Knowledge Distillation from an ensemble of teachers is that the student network will be able to learn better knowledge from multiple teachers as opposed to one, since an ensemble can be more informative, just as how we learn from multiple teachers in school.

First, given an input image, we use an ensemble of models to predict its output segmentation map and average their standard softmaxed prediction logits as in Eqn (1).

Following that, as in [37], we adopt the Kullback-Leibler Divergence to measure the match of the student network's and teacher network's predictions p_1 and p_2 for each pixel (i, j). The KL distance from p_1 to p_2 is computed as

$$\mathcal{L}_{KL}(p_2||p_1) = \sum_{i=1}^{n} \sum_{j=1}^{m} p_2(x_{i,j}) \cdot (\log p_2(x_{i,j}) - \log p_1(x_{i,j})) \tag{2}$$

where n and m are the width and height of the input image. We also adopt the Mean Squared Error (MSE) between p_1 and p_2, computed as

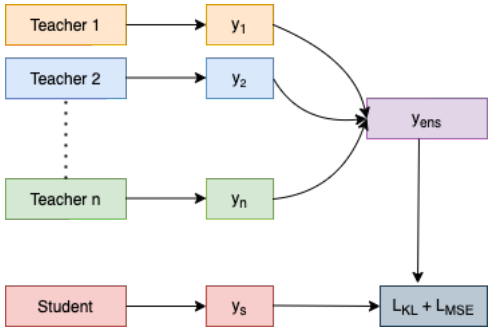

Figure 1. Visualisation of ensemble loss

$$\mathcal{L}_{MSE}(p_1||p_2) = \frac{1}{n}\frac{1}{m} \sum_{i=1}^{n} \sum_{j=1}^{m} (p_1(i, j) - p_2(i, j))^2 \tag{3}$$

The visualisation of our ensemble distillation can be seen in Figure 1, where y_{ens} and y_s are the average softmaxed logits of the ensemble of teachers and softmaxed logits of the student respectively.

The total loss is comprised of four terms:

$$\begin{aligned} \mathcal{L} = & \underbrace{\mathcal{L}_{seg}(s, \hat{s})}_{\text{segmentation loss}} + \underbrace{\alpha \mathcal{L}_{exist}(b, \hat{b})}_{\text{exist loss}} \\ & + \underbrace{\mathcal{L}_{KL}(p_1||p_2) + \beta \mathcal{L}_{MSE}(p_1||p_2)}_{\text{distillation loss}} \end{aligned} \tag{4}$$

where $\mathcal{L}_{seg}(s, \hat{s})$ is the cross-entropy loss between $\hat{s}$, the prediction output segmentation map of our student network and the ground truth segmentation map label s. $\mathcal{L}_{exist}(b, \hat{b})$ is the binary cross-entropy loss between the predicted student network output $\hat{b}$ of the existence of lanes and b is the ground truth of the existence of lanes. $\mathcal{L}_{KL}$ and $\mathcal{L}_{MSE}$ are defined in Eqn (2) and (3) respectively. The parameters α and β balance the influence of exist loss and distillation loss respectively. When training individual models before ensembling, only segmentation loss and exist loss is used.

3.3. Self-training

We combine Knowledge Distillation and self-training by using the model trained with Knowledge Distillation from our ensemble of teachers as the teacher model for self-training which will generate pseudo labels on the CULane dataset. The pseudo labels generated by the teacher model can be soft (probability segmentation map) or hard (segmentation map). We then train a student model with both the labeled data and pseudo labels.

For soft pseudo labels, we train the student model which minimizes the cross-entropy loss between its output and the labeled data, and the MSE and KL loss between the student's output and the teacher's pseudo label, similar to Eqn (4).

For hard pseudo labels, we train the student model which minimizes the combined cross-entropy loss between both labeled and unlabeled data.

3.4. Lane prediction

We use the datasets of TuSimple and CULane. TuSimple is used as our labeled dataset for training, ensemble learning and knowledge distillation from an ensemble of teachers. We ignore the labels provided by CULane and use it as unlabeled data for our self-training experiments. For TuSimple and CULane, in the evaluation stage, we feed the input images into our network. Our network outputs multi-channel probability segmentation maps and a lane existence probability vector $[P_1, P_2, P_3, P_4, P_5, P_6]$ where $P_i, i = 1, .., 6$ depicts the probability of an existence of a lane in its respective spatial positions from left to right for a total of six lanes.

Following [15, 22], we first use a 9 x 9 kernel to smooth the probability maps. For every $P_i > 0.5$, we determine its corresponding probability segmentation map to have a lane and search a total of 56 rows with a y pixel gap of 10 for the pixel x position with the highest probability. Finally, we use cubic splines to connect these positions for the final lane output.

3.5. Architecture Design

The initial ENet [23] model is an encoder-decoder structure comprising of $E_1 \sim E_4$, D_1 and D_2, where E and D are encoders and decoders respectively. We follow the design implementations of [15, 22], and a small network N_1 is added to predict the existence of lanes. The encoder module is shared with N_1 to preserve memory space. Dilated convolution [36] was also added to replace the original convolution layers used in the initial ENet design, for N_1. This was done to increase the receptive field of the network without increasing the number of parameters. Lastly, feature concatenation was also used to fuse the output of E_3 and E_4, so that the output of the encoder can benefit from information encoded in previous layers. This change was made as the resolution of feature maps in the initial design was too small and led to a severe loss of information.

4. Experiments

Datasets. Figure 2 shows a few video frames of the two datasets that we use in our experiments, TuSimple [29] and CULane [22]. TuSimple mainly consists of good weather conditions, while CULane includes video frames taken at night and splits its test dataset into normal and 8 different challenging categories. Many implementations of lane detection have been tested on TuSimple, such as [17, 15, 16, 22]. The size of TuSimple is 3626 frames for the training set and 2782 frames for the testing set. CULane consists of 88880 frames for training set, 9675 for validation and 34680 for testing set. Both CULane and TuSimple has no more than 5 lanes in each video frame.

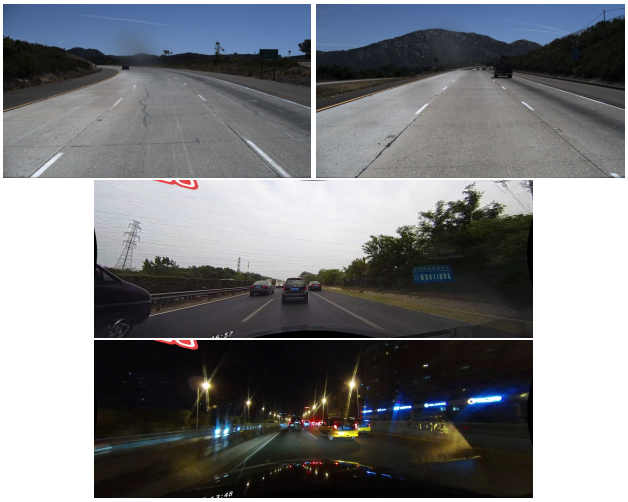

Figure 2. Some TuSimple(1st row) and CULane(2nd & 3rd row) video frames

Evaluation metrics. To compare our results with previous studies with the TuSimple dataset, we use the official metric [29] (accuracy) provided by TuSimple as the evaluation criterion. The official metric also reports false positive (FP) and false negative (FN) values. Accuracy is calculated as $Accuracy = \frac{N_{pred}}{N_{gt}}$, where N_{pred} is the number of correctly predicted lane points and N_{gt} is the number of ground truth lane points.

Implementation details. Following [15, 22], the input images from TuSimple and CULane are resized to 368x640. Stochastic Gradient Descent (SGD) [8] is used to train our models with initial learning rate set to 0.01. A difference from [15] is that we additionally apply learning rate scheduling with poly learning rate rule, with a target learning rate of $1e{-}05$. Batch size is set to 12 and the total number of training epochs is set to 300, or 81600 iterations. The cross-entropy loss of background pixels are multiplied by 0.4. Loss coefficients α and β mentioned in Eqn (4) are set as 0.1 and 0.5 respectively.

Following [15], we also perform data augmentation during training. Random rotation, random cropping and horizontal flipping are used to process the input images before being fed into the network. The same augmentation strategy is applied to single model training, knowledge distillation and self-training experiments. We apply random crop from a size of 373x645 to 368x640. As for rotation, we rotate images between -2 and 2 degrees, and lastly, a horizontal flip probability of 0.5.

We use the base ENet with an additional P_1 network to detect the existence of lanes as our baseline model. We

Figure 3. Visualisation of lane detection with ENet (Ours)

Experiment	Acc	FP	FN
ENet (Theirs)	94.87%	**0.0458**	**0.0387**
ENet (Ours)	**95.05%**	0.0775	0.0603

Table 1. Accuracy, False Positive (FP) and False Negative (FN) results based on TuSimple benchmark for ENet (Theirs) and ENet (Ours).

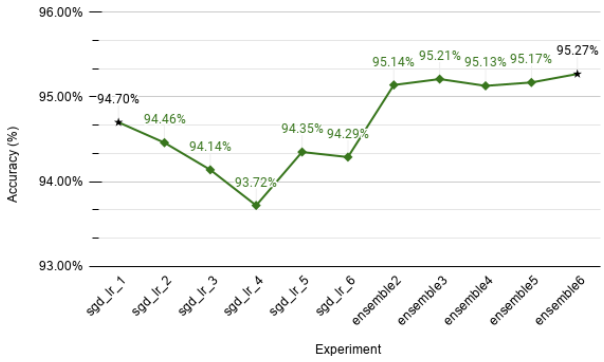

Figure 4. Line graph of respective experiments against accuracy

use the implementation from [15], and port their codebase[1] from torch7 to a PyTorch implementation[2], our framework of choice. This is the first repository for a PyTorch implementation of ENet + P_1 lane detection on TuSimple. Using the optimization settings mentioned above, we were able to reproduce similar results to that of [15], as shown in Table 1. We illustrate some prediction outputs from ENet (Ours) in Figure 3.

4.1. Results

Ensemble learning. Thereafter, we train a total of 6 models of the same architecture and optimization settings as ENet (Ours) and perform ensemble learning on them. Respective epochs are chosen via the best mean Intersection Over Union (mIoU) of the model's predicted outputs and TuSimple validation set labels. Table 2 summarizes the performance of our experiments.

Comparing the accuracy results between ENet (Ours) in Table 1 and sgd_lr_1 $\sim$ sgd_lr_6 in Table 2 which were run with the same optimization settings further derives the point that DNNs are highly flexible algorithms, where even the same settings and dataset will produce different performances each time it is run. This is illustrated in Figure 4.

We observe that even with just an ensemble of two models , the performance gain was 0.44%, peaking at 0.57% for an ensemble of 6 models. This suggests the effectiveness of ensemble learning.

Knowledge Distillation. We show qualitative results that Knowledge Distillation from an ensemble of same-architecrture teachers, trained on the same optimization settings and minimizing Eqn (4), was able to provide very similar performance gains (+0.51% compared to +0.57%) from ensemble learning, without the drawback of incurring any increase in evaluation runtime. Runtime in Table 2 is recorded with a single GPU (NVIDIA Geforce RTX 2080 Ti) by taking the mean evaluation time of all 2782 test samples. We note that the runtime of an ensemble of 6 models is significantly higher compared to just 1. (3ms vs 21ms). This is with a simple averaging ensemble. Should an ensembling method such as Boosting be used, the runtime would be significantly higher. The runtime of our student model ENet-KD is very similar to our baseline model.

This supports our hypothesis that Knowledge Distillation from an ensemble of same-architecture teachers is effective for semantic segmentation and the domain of lane detection. For time sensitive tasks, one may be able to utilize Knowledge Distillation to improve performances instead of ensemble learning, even if the student and teacher is of the same architecture and size.

We illustrate the performance of ENet-KD compared to ENet (Ours); sgd_lr_1 in Figure 5.

Self-training. Table 3 depicts the experimental results of self-training in conjunction with Knowledge Distillation from an ensemble of teachers. ENet-self-1 was conducted using ENet-KD to generate soft pseudo labels for the unlabeled data (CULane), as described in Section 3.3. 1K additional CULane unlabeled data was used in conjunction with the labeled data to train ENet-self-1.

ENet-self-2 was conducted using ENet-KD to generate hard pseudo labels for the unlabeled data, as described in Section 3.3. Similarly, 1K additional unlabeled data from CULane was used. Comparing ENet-self-2 and ENet-self-1, we observe that generating hard pseudo labels and using the combined cross-entropy of labeled and unlabeled data provided better performance.

[1]Codebase: https://github.com/cardwing/Codes-for-Lane-Detection/tree/master/ENet-TuSimple-Torch

[2]Codebase: https://github.com/jwngo/ENet-tuSimple

Experiment	Details	Acc	FP	FN	Acc Gain (+)	FP Gain (-)	FN Gain (-)	Runtime (ms)
ENet (baseline); sgd_lr_1	epoch = 268	94.70%	0.0717	0.064	-	-	-	3.59
ENet (Ours); sgd_lr_2	epoch = 205	94.46%	0.0751	0.0730	-	-	-	3.57
ENet (Ours); sgd_lr_3	epoch = 222	94.14%	0.0804	0.0758	-	-	-	3.66
ENet (Ours); sgd_lr_4	epoch = 244	93.72%	0.0855	0.0839	-	-	-	3.56
ENet (Ours); sgd_lr_5	epoch = 203	94.35%	0.0919	0.0695	-	-	-	3.55
ENet (Ours); sgd_lr_6	epoch = 246	94.29%	0.0921	0.0739	-	-	-	3.57
ensemble2	-	95.14%	0.0645	0.0549	0.44%	-0.00720	-0.00910	6.63
ensemble3	-	95.21%	0.0604	0.0536	0.51%	-0.0113	-0.0104	10.0
ensemble4	-	95.13%	0.0605	0.0558	0.43%	-0.0112	-0.00820	13.2
ensemble5	-	95.17%	0.0603	0.0545	0.47%	-0.0114	-0.00950	16.9
ensemble6	-	**95.27%**	**0.0595**	0.0530	**0.57%**	**-0.0122**	-0.0110	20.9
ENet-KD	epoch = 168	95.21%	0.0718	**0.0529**	0.51%	+0.0001	**-0.0111**	3.58

Table 2. Summary of experimental data. ensembleX states the number of models in the ensemble, $X = \{1, ..., 6\}$ ensemble2 uses sgd_lr_1 and sgd_lr_2, ensemble3 uses sgd_lr_1 $\sim$ sgd_lr_3 and so on. Acc Gain represents the increase in accuracy for that respective ensemble model from the top acc of the single models, sgd_lr_1. FP and FN Gain for respective ensemble results also take sgd_lr_1 as comparison. sgd_lr_1 $\sim$ sgd_lr_6 are all trained with our ENet baseline architecture and settings. ENet-KD refers to our untrained baseline model ran with Knowledge Distillation from an ensemble of all 6 teachers. Runtime calculated based on mean of 2782 sample images on a single NVIDIA Geforce RTX 2080 Ti.

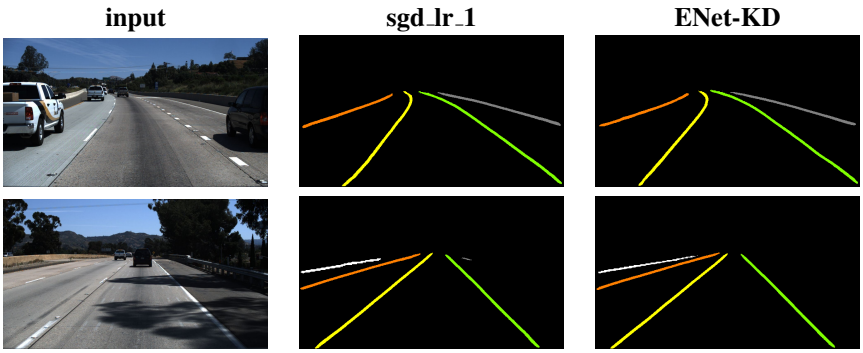

Figure 5. Performance of different algorithms on TuSimple dataset.

Experiment	Details	Acc	FP	FN
ENet-KD	epoch = 168	**95.21%**	**0.0718**	**0.0529**
ENet-self-1	epoch = 168	90.19%	0.1623	0.1817
ENet-self-2	epoch = 189	93.49%	0.0863	0.0990
ENet-self-3	epoch = 163	93.52%	0.0808	0.0916
ENet-self-4	epoch = 230	93.62%	0.0820	0.0879

Table 3. Results of self-training experiments.

ENet-self-3 was conducted similarly to ENet-self-2, but with 3K additional unlabeled data. ENet-self-4 was conducted similarly, but with 10K additional unlabeled data. Increasing the amount of unlabeled data did improve the accuracy of these models, although not by much.

Contrary to our hypothesis that self-training in conjunction with Knowledge Distillation will provide an additional performance boost, adding self-training actually caused a drop in performance. However, we believe that this could be due to self-training being beneficial only with large amount of unlabeled data used. For example, [32] made use of 300M additional unlabeled data.

Due to constraints in time and resources, we were only able to add up to 10K unlabeled images from CULane. Although our preliminary results does show that increasing the amount unlabeled data does indeed increase the performance from using less unlabeled data, it still performs worse than without self-training, even with +10K unlabeled data which quadrupled our training time to over 40 hours.

Test Time Augmentation (TTA). We also conducted experiments on TTA [18]. In order to improve the performance accuracy of our model during evaluation, one of the techniques that we could use is TTA. TTA adds variety to the predictions and has empirically shown improvement in evaluation metrics in [18]. TTA is similar to data augmentation, except that we perform these augmentations during test (evaluation) time, and pass multiple images into the network as input. Then, we aggregate these predictions and get the mean probability segmentation map as our output. In

Experiment	Acc	FP	FN
ENet-KD	**95.21%**	0.0718	**0.0529**
ENet-KD-TTA1	89.40%	0.154	0.193
ENetKD-TTA2	95.03%	**0.0701**	0.0563

Table 4. Summary of results of TTA experiments.

our experiment, we ran TTA with rescaling factors of (0.5, 1.0, 1.5) and horizontal flip. We end up with a total of 6 images passed into the network. Table 4 depicts our experimental results.

We found that we were unable to achieve performance gains from TTA. ENet-KD-TTA1 was conducted with 6 input images as mentioned above. Since we did not achieve any performance gains, we investigated if it was due to issues in rescaling, which might cause information loss for semantic segmentation. To that note, we conduct ENet-KD-TTA2 with only 2 input images, with only a vanilla image and horizontally flipped one. We note that the accuracy was much closer to ENet-KD, with slight improvements in False Positive (FP) rate.

Past research in TTA were mostly done with classification instead of semantic segmentation, and our preliminary results shows that TTA might not be applicable for lane detection. We also experimented with different scaling interpolation methods like nearest neighbours, billinear and whether to align corners when rescaling, with very similar results and hence were not included in the summary.

5. Conclusion

In this work, we present our approach in improving performances in lane detection by making use of otherwise 'wasted' resources of multiple trained same-architecture models, and our approach in performing self-training in conjunction with knowledge distillation, inspired by the generalisability of self-training described in [32]. We propose that for time-sensitive tasks, one can use Knowledge Distillation as an alternative to ensemble learning to achieve very similar performance gains as compared to ensemble learning without any additional unlabeled data or increased evaluation time.

We also consider future directions to extend our work.

1. For a task domain such as lane detection where unlabeled data is relatively easy to obtain, with more time and resources, we may experiment with much larger amounts of unlabeled data to fully utilize self-training's benefits at scale, with some promising preliminary results shown in this paper.

2. For our ensemble learning and knowledge distillation experiments, with more time, we could conduct experiments to determine the optimal number of teachers for distillation to achieve the best performance gains.

3. Our methods mainly utilized the simplest methods of distillation and ensemble learning (simple averaging). With the great amount of attention that distillation is receiving, we may experiment with different methods such as ensemble of features instead of logits, or imitating the teachers' predictions with a gating component to take advantage of the diversity of individual teachers in an ensemble such as in [27]. We could also experiment with different ensemble learning methods, as mentioned in Section 3.1.

4. Compared to ENet-SAD [15], since we did not make any changes to the architecture, while our results improved from our baseline, it is still lower than that provided in ENet-SAD. We could experiment with using an ensemble of pre-trained ENet-SAD as teachers to further increase the performance of ENet-SAD without any increase in evaluation time. Should there be similar improvements in performance ($\mp$0.5%) as reported in this paper, we would outperform the current state-of-the-art lane detection performance for TuSimple reported in [17] with faster evaluation times.

References

[1] Jose M Alvarez, Theo Gevers, Yann LeCun, and Antonio M Lopez. Road scene segmentation from a single image. In *European Conference on Computer Vision*, pages 376–389. Springer, 2012.

[2] Rohan Anil, Gabriel Pereyra, Alexandre Passos, Robert Ormandi, George E Dahl, and Geoffrey E Hinton. Large scale distributed neural network training through online distillation. *arXiv preprint arXiv:1804.03235*, 2018.

[3] Eric Arazo, Diego Ortego, Paul Albert, Noel E. O'Connor, and Kevin McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning, 2020.

[4] Yauhen Babakhin, Artsiom Sanakoyeu, and Hirotoshi Kitamura. Semi-supervised segmentation of salt bodies in seismic images using an ensemble of convolutional neural networks. In *German Conference on Pattern Recognition*, pages 218–231. Springer, 2019.

[5] Massimo Bertozzi and Alberto Broggi. Gold: A parallel real-time stereo vision system for generic obstacle and lane detection. *IEEE transactions on image processing*, 7(1):62–81, 1998.

[6] Christopher M Bishop et al. *Neural networks for pattern recognition*, page 365. Oxford university press, 1995.

[7] Amol Borkar, Monson Hayes, and Mark T Smith. Robust lane detection and tracking with ransac and kalman filter. In *2009 16th IEEE International Conference on Image Processing (ICIP)*, pages 3261–3264. IEEE, 2009.

[8] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[9] Clemens-Alexander Brust, Sven Sickert, Marcel Simon, Erik Rodner, and Joachim Denzler. Convolutional patch networks with spatial prior for road detection and urban scene understanding. *arXiv preprint arXiv:1502.06344*, 2015.

[10] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006.

[11] Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In *Proceedings of the twenty-first international conference on Machine learning*, page 18, 2004.

[12] Richard O Duda and Peter E Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[13] Tommaso Furlanello, Zachary C Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. *arXiv preprint arXiv:1805.04770*, 2018.

[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[15] Yuenan Hou, Zheng Ma, Chunxiao Liu, and Chen Change Loy. Learning lightweight lane detection cnns by self attention distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1013–1021, 2019.

[16] Seokwoo Jung, Sungha Choi, Mohammad Azam Khan, and Jaegul Choo. Towards lightweight lane detection by optimizing spatial embedding, 2020.

[17] Yeongmin Ko, Jiwon Jun, Donghwuy Ko, and Moongu Jeon. Key points estimation and point instance segmentation approach for lane detection. *arXiv preprint arXiv:2002.06604*, 2020.

[18] Tianqi Liu and Qizhan Shao. Bert for large-scale video segment classification with test-time augmentation, 2019.

[19] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[20] Seyed-Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. *arXiv preprint arXiv:1902.03393*, 2019.

[21] Rahul Mohan. Deep deconvolutional networks for scene parsing. *arXiv preprint arXiv:1411.4101*, 2014.

[22] Xingang Pan, Jianping Shi, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Spatial as deep: Spatial cnn for traffic scene understanding. *arXiv preprint arXiv:1712.06080*, 2017.

[23] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016.

[24] Zequn Qin, Huanyu Wang, and Xi Li. Ultra fast structure-aware deep lane detection. *arXiv preprint arXiv:2004.11757*, 2020.

[25] Ellen Riloff. Automatically generating extraction patterns from untagged text. In *Proceedings of the national conference on artificial intelligence*, pages 1044–1049, 1996.

[26] Ellen Riloff and Janyce Wiebe. Learning extraction patterns for subjective expressions. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 105–112, 2003.

[27] Sebastian Ruder, Parsa Ghaffari, and John G Breslin. Knowledge adaptation: Teaching to adapt. *arXiv preprint arXiv:1702.02052*, 2017.

[28] H Scudder. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11(3):363–371, 1965.

[29] TuSimple. Tusimple benchmark. `https://github.com/TuSimple/tusimple-benchmark`.

[30] Meng-Chieh Wu, Ching-Te Chiu, and Kun-Hsuan Wu. Multi-teacher knowledge distillation for compressed video action recognition on deep neural networks. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2202–2206. IEEE, 2019.

[31] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. Unsupervised data augmentation for consistency training, 2020.

[32] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10687–10698, 2020.

[33] Ze Yang, Linjun Shou, Ming Gong, Wutao Lin, and Daxin Jiang. Model compression with two-stage multi-teacher knowledge distillation for web question answering system. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 690–698, 2020.

[34] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*, pages 189–196, 1995.

[35] Shan You, Chang Xu, Chao Xu, and Dacheng Tao. Learning from multiple teacher networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1285–1294, 2017.

[36] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[37] Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. Deep mutual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4320–4328, 2018.

[38] Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin D Cubuk, and Quoc V Le. Rethinking pre-training and self-training. *arXiv preprint arXiv:2006.06882*, 2020.